

Eclipse And Java For Total Beginners Companion Tutorial Document

By Mark Dexter

Table of Contents

Table of Contents.	1
Introduction.	2
Tutorial Target Audience.	2
Tutorial Objectives.	2
Why learn Java with Eclipse?.	2
Topics Covered.	3
Tutorial Approach.	3
Getting The Most From This Tutorial	3
Downloading and Installing Eclipse.	3
Lesson Outlines.	4
Lesson 1 – Create Your First Java Class.	4
Lesson 2 – Add Methods To Class.	4
Lesson 3 – Use Eclipse Scrapbook.	5
Lesson 4 – JUnit Testing in Eclipse, Part 1.	5
Lesson 5 – JUnit Testing Continued.	5
Lesson 6 – Using Test-First Development in Eclipse.	6
Lesson 7 – Create Book Class.	6
Lesson 8 – Add Person to Book Class.	6
Lesson 9 – MyLibrary Class and ArrayList	6
Lesson 10 – Start on MyLibrary Class.	6
Lesson 11 – Create first methods in MyLibrary class.	6
Lesson 12 – Create checkOut, checkIn Methods.	6
Lesson 13 – Continue checkOut Method.	6

Eclipse and Java for Total Beginners

Tutorial Companion Document

Lesson 14 – Finish checkOut Method.	7
Lesson 15 – Finish MyLibrary Methods.	7
Lesson 16 – Create main Method and JAR File.	7
Glossary of Terms.	7
Additional Resources.	10
Eclipse Websites.	10
Java Websites.	10
Java Books.	10
Code Snapshots.	11
Lesson 1 – Person class (fields and constructor)	11
Lesson 2 – Person class (with get and set methods)	11
Lesson 3 (scrapbook snapshots)	12
Lesson 5 – PersonTest class.	12
Lesson 6 – Added toString() Method to Person class.	13
Lesson 7 – TestBook and Book Classes.	14
Lesson 8 – Add Person to Book Class.	15
Lesson 9 – MyLibrary Class and ArrayList	16
Lesson 10 – Start on MyLibrary Class.	16
Lesson 11 – Create first methods in MyLibrary class.	17
Lesson 12 – Create checkOut, checkIn Methods.	19
Lesson 13 – Continue checkOut Method.	22
Lesson 14 – Finish checkOut Method.	24
Lesson 15 – Finish MyLibrary Methods.	25
Lesson 16 – Create main Method and JAR File.	30

Introduction

This document is designed to accompany the “Eclipse And Java For Total Beginners” video tutorial, which is available at <http://www.eclipse.org/home/newcomers.php>.

Tutorial Target Audience

This tutorial is targeted for people who are new to Eclipse and to Java. It is designed to work either for those with prior programming experience in other languages or for those without prior experience.

Tutorial Objectives

The objectives of this tutorial are as follows:

- Demonstrate the basics of using Eclipse for writing Java programs
- Demonstrate how to use Eclipse for agile software development
- Demonstrate how to use existing Java learning resources (tutorials, examples, books) within Eclipse
- Create a foundation for continuing to learn Java and Eclipse

Note that Java and Eclipse are both large subject areas that cannot possibly be mastered in a short period of time. This tutorial will help get you started and give you some of the skills needed to learn on your own.

Why learn Java with Eclipse?

There are many ways to learn how to program in Java. The author believes that there are advantages to learning Java using the Eclipse integrated development environment (IDE). Some of these are listed below:

- Eclipse provides a number of aids that make writing Java code much quicker and easier than using a text editor. This means that you can spend more time learning Java, and less time typing and looking up documentation.
- The Eclipse debugger and scrapbook allow you to look inside the execution of the Java code. This allows you to “see” objects and to understand how Java is working behind the scenes
- Eclipse provides full support for agile software development practices such as test-driven development and refactoring. This allows you to learn these practices as you learn Java.
- If you plan to do software development in Java, you’ll need to learn Eclipse or some other IDE. So learning Eclipse from the start will save you time and effort.

The chief concern with learning Java with an IDE is that learning the IDE itself will be difficult and will distract you from learning Java. It is hoped that this tutorial will make learning the basics of Eclipse relatively painless so you can focus on learning Java.

Topics Covered

This tutorial will cover the following topics:

- Basics of Eclipse for Java development
- Basics of Java and object-oriented programming (OOP)
- Test-driven development (TDD) and unit testing in Eclipse

Tutorial Approach

The tutorial is organized around the following activities.

- Write a small sample Java application to track your personal lending library.
- Use the “test-first” approach to develop most methods.
- Write a “test drive” program and create an executable JAR file, and run the JAR file from the system console.

Concepts are introduced as needed during the development of the sample applications.

Getting The Most From This Tutorial

This tutorial can be used as an in-depth demonstration of Java development in Eclipse. However, if you want to actually learn how to write Java programs in Eclipse, the following approach is recommended:

- Have Eclipse installed and ready to go.
- Work side-by-side with the lessons, pausing and rewinding as needed.
- Use this guide as needed.
- Consult other resources as needed to understand the topics covered in greater depth.
- Keep a positive attitude!

Downloading and Installing Eclipse

Before Installing Eclipse, you need to have either the Java JDK (Java development kit) or Java JRE (Java runtime engine) installed on your computer. These are available at <http://java.sun.com/javase/downloads/index.jsp>.

Installing the JDK or JRE is reasonably simple. Detailed, step-by-step instructions, if needed, are available in the PDF Eclipse Tutorial at the

Eclipse and Java for Total Beginners

Tutorial Companion Document

<https://www.arctechsoftware.com/tutorial/welcomePage.do>. (Follow the link to “Beginning Eclipse”.)

For Java development, the JDK is recommended because it allows you to see documentation and source code for the standard Java classes. However, either the JDK or JRE will work for this tutorial.

This tutorial is based on Eclipse 3.3, although you could use 3.2 and probably later versions as well. Here are the steps to install Eclipse 3.3 from www.eclipse.org:

- Navigate to www.eclipse.org/downloads
- Select “Eclipse IDE for Java Developers”. If your platform is Linux or MacOSX, be sure to select the link to the right. Note that you can use “Eclipse IDE for Java EE Developers”, “Eclipse for RCP/Plug-in Developers”, or “Eclipse Classic” as well. All of these include the Java development portions of Eclipse used in this tutorial.
- On the www.eclipse.org/downloads page, follow the link “Find out more”. Scroll your browser to display the far right-hand side of the screen to the column “Tutorials and Help”. The first tutorial is a Screencam tutorial that steps you through downloading and installing Eclipse on Windows.

The Eclipse installation is very straightforward. There is no installation program. Instead, you just create the top-level folder and then unzip the file inside this folder. In Windows XP, for example, just copy the zip file to your root directory (e.g., “C:\”) and then unzip the downloaded zip file. This will create a folder called “C:\eclipse”. The Eclipse programs will be created in several subfolders (configuration, features, plugins, readme). The procedure for Linux is similar, except you unzip the .tar.gz file.

Lesson Outlines

Lesson 1 – Create Your First Java Class

- Create Java project in Eclipse
- Create Java package
- Introduce classes and objects, naming conventions
- Write a simple Java class (Person)

Lesson 2 – Add Methods To Class

- Introduce Eclipse Views and Perspectives
- Introduce Eclipse user interface – drag / drop, context menus, help

- Add get and set methods to Person class

Lesson 3 – Use Eclipse Scrapbook

- Introduce Eclipse Scrapbook
- Introduce Java expressions, statements
- Discuss Java packages
- Create Person object in Scrapbook

Lesson 4 – JUnit Testing in Eclipse, Part 1

- Create test source folder
- Create PersonTest class
- Run first JUnit test

Lesson 5 – JUnit Testing Continued

- Test Person class – part 2
- Create test methods for constructor, getName, and getMaximumBooks

Lesson 6 – Using Test-First Development in Eclipse

- Use test-first approach to write the Person toString() method

Lesson 7 – Create Book Class

- Create BookTest before creating Book class
- Test Book constructor
- Create get and set methods

Lesson 8 – Add Person to Book Class

- Create a relationship between the Book class and the Person Class
- Test getPerson method
- Create JUnit Test Suite

Lesson 9 – MyLibrary Class and ArrayList

- How can we hold books, etc. in a collection?
- MyLibrary object to hold Person & Entry objects
- Introduce ArrayList in scrapbook

Lesson 10 – Start on MyLibrary Class

- Create MyLibraryTest JUnit test
- Create testMyLibrary to test MyLibrary constructor
- Create MyLibrary constructor

Lesson 11 – Create first methods in MyLibrary class

- Create test method for addBook, removeBook methods
- Create addBook, removeBook methods and test
- Create addPerson, removePerson methods

Lesson 12 – Create checkOut, checkIn Methods

- Create test for checkOut, checkIn methods
- Write checkout method
- Introduce if / then / else syntax
- Introduce boolean method
- Write checkIn method

Lesson 13 – Continue checkOut Method

- Test checkOut, checkIn methods
- Fix compiler error – misplaced {}
- Add test for maximum books
- Create test for getBooksForPerson() method

Lesson 14 – Finish checkOut Method

- Write getBooksForPerson method
- Introduce “for each” loop
- Introduce logical and operator “&&”
- NullPointerException errors
- Complete checkOut Method

Lesson 15 – Finish MyLibrary Methods

- Create test for getAvailableBooks
- Create getAvailableBooks

- Create getUnavailableBooks

Lesson 16 – Create main Method and JAR File

- Introduce main method
- Write a main method
- Run MyLibrary as Java application
- Export to JAR file and run from Windows

Glossary of Terms

<u>Term</u>	<u>Quick Definition</u>
Access Modifier	Reserved words “public”, “private”, “protected” in Java. Control whether classes and members may be accessed from any class, only this class, subclasses. Default is access from any class in the package.
Agile (or Extreme) Development	Methodology for developing software that emphasizes, among other things, unit testing as part of development process.
API (Application Programming Interface)	The way one program uses another program. In Java, the API can be thought of as the collection of public methods for a class or package.
Class	Main building block in Java. Contains members, including fields and methods. Classes are the “blueprint” for creating objects.
Constructor	Special block of code used to create an instance of a class (or, if you prefer, an object whose type is the class). Used with the “new” keyword (e.g., <code>Person p = new Person()</code> calls the <code>Person()</code> constructor).
Field	Member in a class that holds data (e.g., name, age, etc.). Usually marked private so that other programs cannot directly access.
IDE (Integrated Development Environment)	Program, like Eclipse, that provides the different tools required to develop a software package.
JVM (Java Virtual Machine) (also known as Java Runtime Engine or JRE)	The program that runs Java programs on a specific platform. Java source code is compiled into .class files. These contain the instructions used by the JVM to actually run the programs on a Windows PC, a Linux computer, a Mac computer, etc. The JVM is written for each platform supported by

Eclipse and Java for Total Beginners
Tutorial Companion Document

<u>Term</u>	<u>Quick Definition</u>
	Java.
JUnit Test	A Java class used to test individual methods in a class. Used to build test cases, e.g., when using agile development methodology.
Method	Member in a class that does some processing (e.g., like a subroutine or function in other languages).
Method Argument, MethodParameter	<p>Parameters refers to the list of variables in a method declaration. Arguments are the actual values that are passed in when the method is invoked. When you invoke a method, the arguments used must match the declaration's parameters in type and order. For example, in the method</p> <pre>public setName(String name) {...}</pre> <p>“name” is the parameter for this method. If this method is used as follows:</p> <pre>myObject.setName("Fred");</pre> <p>“Fred” is the argument of the method and it must match the type of the method’s parameter.</p>
Method Signature	A method’s name plus it’s parameter list. For example, a method defined as “setName (String name)” has a method signature of “setName(String)”. Method signatures are important because they allow methods to be overloaded (i.e., have the same name but different signatures). For example, the method “setName(String firstName, String lastName) could be an overload of “setName(String name)” because it as a different signature (“setName(String, String)”).
Object	An instance of a class. For example, Cookie could be a class, and a cookie (e.g. “thisCookie”) would be an object created using the class. In other words, “thisCookie” is an object of type Cookie or an instance of Cookie.
Overload (Method)	To provide multiple methods with the same name but different parameters (i.e., same name but different signatures).
Override (Method)	When a subclass implements a method inherited from the super class, this method is said to be overridden.
Package	Packages are imported into a source file to save typing the full name of the class (e.g., can say “Person” instead of

Eclipse and Java for Total Beginners
Tutorial Companion Document

<u>Term</u>	<u>Quick Definition</u>
	“org.eclipse.training.librarytutorial.Person” and to avoid the possibility of two classes having identical names.
Project	In Eclipse, a way to organize your work. An Eclipse workspace can contain multiple projects. Each project can contain multiple packages. Each package can contain multiple classes.
Refactor	To improve a program without changing the way it works (i.e., its API). Example include renaming fields or variables, streamlining code, etc. Very important in agile development because of emphasis on self-documenting code.
Reference Variable	In Java, variable that holds an object reference (e.g., <code>p = new Person();</code>). Points to an area on the “heap” where the object resides. Contrast with value variable.
Scrapbook Page	Area in Eclipse where you can execute Java code “snippets” and see how they work. Great for experimenting with Java statements.
Static Method	A method that belongs to the entire class instead of one instance of the class. Invoked with <code><Class>.<Method></code> (e.g., <code>Person.getTotalCount()</code>). Used for methods that don’t rely on any one instance of a class.
Swing	A set of standard Java packages that implement a graphical user interface without using any “native” code.
SWT (Standard Widget Toolkit)	Set of Java classes and native programs developed by Eclipse to allow Java programs to have the look and feel of native programs on each platform. Used to create the Eclipse IDE.
Type	In Java, an attribute of a variable to indicate either a primitive type (int, boolean, etc.) or class membership. For objects, the type is the class to which it belongs. Types also include interfaces and enumerations.
Value Variable	In Java, variable that holds the value of a Java primitive (e.g., integer, character, etc.). Held in the memory stack. Contrast with reference variable.
Workspace	Top-level container for Eclipse work. Holds multiple projects. In a single Eclipse session, only one workspace can be active.

Additional Resources

There are many resources available for learning more about Eclipse and Java. These are just a few that I've found helpful.

Eclipse Websites

- www.eclipse.org/resources. This lists a number of articles, books, presentations, demonstrations and other resources on a variety of topics related to Eclipse.
- eclipse.newcomer newsgroup. This is a friendly, active newsgroup where newcomers to Eclipse can ask questions. The search feature of this and other newsgroups can be especially valuable, since there is a good chance that your question has already been asked and answered.
- Beginning Eclipse Tutorial on ArcTech Software LLC website. Written tutorial to get you started with Eclipse and Java. Login required to download. It has a very good section on downloading and installing the Java JDK. Link to tutorial is <https://www.arctechsoftware.com/tutorial/tutorial.do?subcatId=1>. Link to home page is <https://www.arctechsoftware.com/tutorial/welcomePage.do>.

Java Websites

- The Java Tutorials from Sun (<http://java.sun.com/docs/books/tutorial/java/index.html>). The gold standard for learning Java, and it's free.
- JavaRanch Big Moose Saloon web site (<http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?category=1>). This has a variety of forums, including Java in General (beginner), Java in General (intermediate), and many other Java topics. Very active and friendly, with knowledgeable moderators.
- The Java Developers Almanac 1.4 (<http://www.exampledepot.com/>). Contains Java code samples for many topics.

Java Books

- Head First Java, by Kathy Sierra & Bert Bates. Excellent, fun, creative book for Java and OOP beginners.
- Thinking In Java, by Bruce Eckel. Excellent, thorough reference for Java. For all levels of programmer.
- Effective Java, by Joshua Bloch. Concise book documents specific recommendations for Java best practices. For intermediate to advanced programmers.

Code Snapshots

The following pages contain code snapshots as of the end of each lesson. These can be used to compare your code to or to help you fix any problems you might have. If needed, you can copy and paste this code into your Eclipse Java source files. Also, if you want to start the tutorial in the middle, these can help you catch up to the correct point.

Lesson 1 – Person class (fields and constructor)

```
package org.totalbeginner.tutorial;

public class Person {
    // fields
    private String name; // name of the person
    private int maximumBooks; // most books the person can check out

    // constructors
    public Person() {
        name = "unknown name";
        maximumBooks = 3;
    }
}
```

Lesson 2 – Person class (with get and set methods)

Note: Highlighted code added in this lesson.

```
package org.totalbeginner.tutorial;

public class Person {
    // fields
    private String name; // name of the person
    private int maximumBooks; // most books the person can check out

    // constructors
    public Person() {
        name = "unknown name";
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        maximumBooks = 3;
    }

    //methods
    public String getName() {
        return name;
    }

    public void setName(String anyName) {
        name = anyName;
    }

    public int getMaximumBooks() {
        return maximumBooks;
    }

    public void setMaximumBooks(int maximumBooks) {
        this.maximumBooks = maximumBooks;
    }
}
```

Lesson 3 (scrapbook snapshots)

Expressions:

```
2 + 2
```

```
int a = 5;
a = a * 10;
a
```

(after import of org.totalbeginner.tutorial.*)

```
Person p = new Person();
p.setName("Fred");
P
```

Lesson 5 – PersonTest class

```
package org.totalbeginner.tutorial;

import junit.framework.TestCase;

public class PersonTest extends TestCase {

    public void testPerson() {
        Person p1 = new Person();
        assertEquals("unknown name", p1.getName());
        assertEquals(3, p1.getMaximumBooks());
    }

    public void testSetName() {
        Person p2 = new Person();
        p2.setName("Fred");
        assertEquals("Fred", p2.getName());
    }

    public void testSetMaximumBooks() {
        Person p3 = new Person();
        p3.setMaximumBooks(10);
        assertEquals(10, p3.getMaximumBooks());
    }
}
```

Lesson 6 – Added toString() Method to Person class

Note: Person and PersonTest classes are complete at this point.

```
package org.totalbeginner.tutorial;

public class Person {
    // fields
    private String name; // name of the person
    private int maximumBooks; // most books the person can check out

    // constructors
    public Person() {
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        name = "unknown name";
        maximumBooks = 3;
    }

    //methods
    public String getName() {
        return name;
    }

    public void setName(String anyName) {
        name = anyName;
    }

    public int getMaximumBooks() {
        return maximumBooks;
    }

    public void setMaximumBooks(int maximumBooks) {
        this.maximumBooks = maximumBooks;
    }

    public String toString() {
        return this.getName() + " (" + this.getMaximumBooks() +
            " books)";
    }
}
```

```
package org.totalbeginner.tutorial;

import junit.framework.TestCase;

public class PersonTest extends TestCase {

    public void testPerson() {
        Person p1 = new Person();
        assertEquals("unknown name", p1.getName());
        assertEquals(3, p1.getMaximumBooks());
    }

    public void testSetName() {
        Person p2 = new Person();
        p2.setName("Fred");
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        assertEquals("Fred", p2.getName());
    }

    public void testSetMaximumBooks() {
        Person p3 = new Person();
        p3.setMaximumBooks(10);
        assertEquals(10, p3.getMaximumBooks());
    }

    public void testToString() {
        Person p4 = new Person();
        p4.setName("Fred Flintstone");
        p4.setMaximumBooks(7);
        String testString = "Fred Flintstone (7 books)";
        assertEquals(testString, p4.toString());
    }
}
```

Lesson 7 – TestBook and Book Classes

```
package org.totalbeginner.tutorial;

import junit.framework.TestCase;

public class BookTest extends TestCase {

    public void testBook() {
        Book b1 = new Book("Great Expectations");
        assertEquals("Great Expectations", b1.title);
        assertEquals("unknown author", b1.author);
    }
}
```

```
package org.totalbeginner.tutorial;

public class Book {
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
String title;
String author;

public Book(String string) {
    this.title = string;
    this.author = "unknown author";
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public String getTitle() {
    return title;
}
}
```

```
package org.totalbeginner.tutorial;

import junit.framework.Test;
import junit.framework.TestSuite;

public class AllTests {

    public static Test suite() {
        TestSuite suite = new TestSuite("Test for org.totalbeginner.tutorial");
        //$JUnit-BEGIN$
        suite.addTestSuite(BookTest.class);
        suite.addTestSuite(PersonTest.class);
        //$JUnit-END$
        return suite;
    }
}
```

Lesson 8 – Add Person to Book Class

Note: BookTest and Book classes are complete at this point.

```
package org.totalbeginner.tutorial;

public class Book {

    String title;
    String author;
    Person person;

    public Book(String string) {
        this.title = string;
        this.author = "unknown author";
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public void setPerson(Person p2) {
        this.person = p2;
    }

    public Person getPerson() {
        return this.person;
    }
}
```

```
package org.totalbeginner.tutorial;
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
import junit.framework.TestCase;

public class BookTest extends TestCase {

    public void testBook() {
        Book b1 = new Book("Great Expectations");
        assertEquals("Great Expectations", b1.title);
        assertEquals("unknown author", b1.author);
    }

    public void testGetPerson() {
        Book b2 = new Book("War And Peace");
        Person p2 = new Person();
        p2.setName("Elvis");

        // method to say book is loaned to this person
        b2.setPerson(p2);

        // get the name of the person who has the book
        // Person testPerson = b2.getPerson();
        // String testName = testPerson.getName();

        String testName = b2.getPerson().getName();
        assertEquals("Elvis", testName);
    }
}
```

Lesson 9 – MyLibrary Class and ArrayList

Scrapbook Example

```
ArrayList<Book> list = new ArrayList<Book>();

Book b1 = new Book("Great Expectations");
Book b2 = new Book("War and Peace");
list.add(b1);
list.add(b2);

Person p1 = new Person();
p1.setName("Fred");
b1.setPerson(p1);

list.remove(b1);
```

list

Lesson 10 – Start on MyLibrary Class

Note: From this point on, if needed use the Person and PersonTest classes from Lesson 6 and the Book and BookTest classes from Lesson 8.

```
package org.totalbeginner.tutorial;

import java.util.ArrayList;

import junit.framework.TestCase;

public class MyLibraryTest extends TestCase {

    // test constructor
    public void testMyLibrary() {
        MyLibrary ml = new MyLibrary("Test");

        assertEquals("Test", ml.name);

        assertTrue(ml.books instanceof ArrayList);
        assertTrue(ml.people instanceof ArrayList);

    }
}
```

```
package org.totalbeginner.tutorial;

import java.util.ArrayList;

public class MyLibrary {

    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
        books = new ArrayList<Book>();
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        people = new ArrayList<Person>();  
    }  
}
```

Lesson 11 – Create first methods in MyLibrary class

```
package org.totalbeginner.tutorial;  
  
import java.util.ArrayList;  
  
import junit.framework.TestCase;  
  
public class MyLibraryTest extends TestCase {  
  
    private Book b1;  
    private Book b2;  
    private Person p1;  
    private Person p2;  
    private MyLibrary ml;  
  
    // test constructor  
    public void testMyLibrary() {  
        MyLibrary ml = new MyLibrary("Test");  
  
        assertEquals("Test", ml.name);  
  
        assertTrue(ml.books instanceof ArrayList);  
        assertTrue(ml.people instanceof ArrayList);  
    }  
  
    public void setup() {  
        b1 = new Book("Book1");  
        b2 = new Book("Book2");  
        p1 = new Person();  
        p2 = new Person();  
        p1.setName("Fred");  
        p2.setName("Sue");  
        ml = new MyLibrary("Test");  
    }  
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
}  
  
public void testAddBook() {  
    //create test objects  
    setup();  
  
    //test initial size is 0  
    assertEquals(0, ml.getBooks().size());  
  
    ml.addBook(b1);  
    ml.addBook(b2);  
  
    assertEquals(2, ml.getBooks().size());  
    assertEquals(0, ml.getBooks().indexOf(b1));  
    assertEquals(1, ml.getBooks().indexOf(b2));  
  
    ml.removeBook(b1);  
    assertEquals(1, ml.getBooks().size());  
    assertEquals(0, ml.getBooks().indexOf(b2));  
  
    ml.removeBook(b2);  
    assertEquals(0, ml.getBooks().size());  
}  
}
```

```
package org.totalbeginner.tutorial;  
  
import java.util.ArrayList;  
  
public class MyLibrary {  
  
    String name;  
    ArrayList<Book> books;  
    ArrayList<Person> people;  
  
    public MyLibrary(String name) {  
        this.name = name;  
        books = new ArrayList<Book>();  
        people = new ArrayList<Person>();  
    }  
  
    public String getName() {
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        return name;
    }

    public ArrayList<Book> getBooks() {
        return books;
    }

    public ArrayList<Person> getPeople() {
        return people;
    }

    public void addBook(Book b1) {
        this.books.add(b1);
    }

    public void removeBook(Book b1) {
        this.books.remove(b1);
    }

    public void addPerson(Person p1) {
        this.people.add(p1);
    }

    public void removePerson(Person p1) {
        this.people.remove(p1);
    }
}
```

Lesson 12 – Create checkOut, checkIn Methods

```
package org.totalbeginner.tutorial;

import java.util.ArrayList;

import junit.framework.TestCase;

public class MyLibraryTest extends TestCase {
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
private Book b1;
private Book b2;
private Person p1;
private Person p2;
private MyLibrary ml;

// test constructor
public void testMyLibrary() {
    MyLibrary ml = new MyLibrary("Test");

    assertEquals("Test", ml.name);

    assertTrue(ml.books instanceof ArrayList);
    assertTrue(ml.people instanceof ArrayList);

}

public void setup() {
    b1 = new Book("Book1");
    b2 = new Book("Book2");
    p1 = new Person();
    p2 = new Person();
    p1.setName("Fred");
    p2.setName("Sue");

    ml = new MyLibrary("Test");

}

public void testAddBook() {
    //create test objects
    setup();

    //test initial size is 0
    assertEquals(0, ml.getBooks().size());

    ml.addBook(b1);
    ml.addBook(b2);

    assertEquals(2, ml.getBooks().size());
    assertEquals(0, ml.getBooks().indexOf(b1));
    assertEquals(1, ml.getBooks().indexOf(b2));

    ml.removeBook(b1);
    assertEquals(1, ml.getBooks().size());
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
    assertEquals(0, ml.getBooks().indexOf(b2));

    ml.removeBook(b2);
    assertEquals(0, ml.getBooks().size());

}

private void addItem() {
    ml.addBook(b1);
    ml.addBook(b2);
    ml.addPerson(p1);
    ml.addPerson(p2);
}

public void testCheckOut() {
    // set up objects
    setup();
    addItem();

    assertTrue("Book did not check out correctly",
        ml.checkOut(b1,p1));

    assertEquals("Fred", b1.getPerson().getName());

    assertFalse("Book was already checked out",
        ml.checkOut(b1,p2));

    assertTrue("Book check in failed", ml.checkIn(b1));

    assertFalse("Book was already checked in", ml.checkIn(b1));

    assertFalse("Book was never checked out", ml.checkIn(b2));
}

}
```

```
package org.totalbeginner.tutorial;
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
import java.util.ArrayList;

public class MyLibrary {

    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
        books = new ArrayList<Book>();
        people = new ArrayList<Person>();
    }

    public String getName() {
        return name;
    }

    public ArrayList<Book> getBooks() {
        return books;
    }

    public ArrayList<Person> getPeople() {
        return people;
    }

    public void addBook(Book b1) {
        this.books.add(b1);
    }

    public void removeBook(Book b1) {
        this.books.remove(b1);
    }

    public void addPerson(Person p1) {
        this.people.add(p1);
    }

    public void removePerson(Person p1) {
        this.people.remove(p1);
    }

    public boolean checkOut(Book b1, Person p1) {
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        if ( (b1.getPerson() == null) ){
            b1.setPerson(p1);
            return true;
        }
        else {
            return false;
        }
    }

    public boolean checkIn(Book b1) {
        if (b1.getPerson() != null) {
            b1.setPerson(null);
            return true;
        }
        else {
            return false;
        }
    }
}
```

Lesson 13 – Continue checkOut Method

```
package org.totalbeginner.tutorial;

import java.util.ArrayList;

public class MyLibrary {

    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
        books = new ArrayList<Book>();
        people = new ArrayList<Person>();
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
public String getName() {
    return name;
}

public ArrayList<Book> getBooks() {
    return books;
}

public ArrayList<Person> getPeople() {
    return people;
}

public void addBook(Book b1) {
    this.books.add(b1);
}

public void removeBook(Book b1) {
    this.books.remove(b1);
}

public void addPerson(Person p1) {
    this.people.add(p1);
}

public void removePerson(Person p1) {
    this.people.remove(p1);
}

public boolean checkOut(Book b1, Person p1) {
    if ((b1.getPerson() == null) ){
        b1.setPerson(p1);
        return true;
    }
    else {
        return false;
    }
}

public boolean checkIn(Book b1) {
    // TODO Auto-generated method stub
    return false;
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
public ArrayList<Book> getBooksForPerson(Person p1) {  
    // TODO Auto-generated method stub  
    return null;  
}  
  
}
```

Lesson 14 – Finish checkOut Method

```
package org.totalbeginner.tutorial;  
  
import java.util.ArrayList;  
  
public class MyLibrary {  
  
    String name;  
    ArrayList<Book> books;  
    ArrayList<Person> people;  
  
    public MyLibrary(String name) {  
        this.name = name;  
        books = new ArrayList<Book>();  
        people = new ArrayList<Person>();  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public ArrayList<Book> getBooks() {  
        return books;  
    }  
  
    public ArrayList<Person> getPeople() {  
        return people;  
    }  
  
    public void addBook(Book b1) {  
        this.books.add(b1);  
    }  
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
}

public void removeBook(Book b1) {
    this.books.remove(b1);
}

public void addPerson(Person p1) {
    this.people.add(p1);
}

public void removePerson(Person p1) {
    this.people.remove(p1);
}

public boolean checkOut(Book b1, Person p1) {
    int booksOut = this.getBooksForPerson(p1).size();
    if ((b1.getPerson() == null) &&
        booksOut < p1.getMaximumBooks()){
        b1.setPerson(p1);
        return true;
    }
    else {
        return false;
    }
}

public boolean checkIn(Book b1) {
    if (b1.getPerson() != null) {
        b1.setPerson(null);
        return true;
    }
    else {
        return false;
    }
}

public ArrayList<Book> getBooksForPerson(Person p1) {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if ((aBook.getPerson() != null) &&
            (aBook.getPerson().getName()
                .equals(p1.getName())))
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        {  
            result.add(aBook);  
        }  
    }  
    return result;  
}
```

Lesson 15 – Finish MyLibrary Methods

```
package org.totalbeginner.tutorial;  
  
import java.util.ArrayList;  
  
import junit.framework.TestCase;  
  
public class MyLibraryTest extends TestCase {  
  
    private Book b1;  
    private Book b2;  
    private Person p1;  
    private Person p2;  
    private MyLibrary ml;  
  
    // test constructor  
    public void testMyLibrary() {  
        MyLibrary ml = new MyLibrary("Test");  
  
        assertEquals("Test", ml.name);  
  
        assertTrue(ml.books instanceof ArrayList);  
        assertTrue(ml.people instanceof ArrayList);  
    }  
  
    public void setup() {  
        b1 = new Book("Book1");  
        b2 = new Book("Book2");  
        p1 = new Person();  
        p2 = new Person();  
    }  
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
p1.setName("Fred");
p2.setName("Sue");

ml = new MyLibrary("Test");

}

public void testAddBook() {
    //create test objects
    setup();

    //test initial size is 0
    assertEquals(0, ml.getBooks().size());

    ml.addBook(b1);
    ml.addBook(b2);

    assertEquals(2, ml.getBooks().size());
    assertEquals(0, ml.getBooks().indexOf(b1));
    assertEquals(1, ml.getBooks().indexOf(b2));

    ml.removeBook(b1);
    assertEquals(1, ml.getBooks().size());
    assertEquals(0, ml.getBooks().indexOf(b2));

    ml.removeBook(b2);
    assertEquals(0, ml.getBooks().size());

}

public void testCheckOut() {
    // set up objects
    setup();

    addItem();

    assertTrue("Book did not check out correctly",
        ml.checkOut(b1,p1));

    assertEquals("Fred", b1.getPerson().getName());

    assertFalse("Book was already checked out",
        ml.checkOut(b1,p2));
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
    assertTrue("Book check in failed", ml.checkIn(b1));

    assertFalse("Book was already checked in", ml.checkIn(b1));

    assertFalse("Book was never checked out", ml.checkIn(b2));

    // additional test for maximumBooks
    setup();
    p1.setMaximumBooks(1);
    addItem();

    assertTrue("First book did not check out",
        ml.checkOut(b2, p1));
    assertFalse("Second book should not have checked out",
        ml.checkOut(b1, p1));
}

private void addItem() {
    ml.addBook(b1);
    ml.addBook(b2);
    ml.addPerson(p1);
    ml.addPerson(p2);
}

public void testGetBooksForPerson() {
    setup();
    addItem();
    assertEquals(0, ml.getBooksForPerson(p1).size());

    ml.checkOut(b1, p1);

    ArrayList<Book> testBooks = ml.getBooksForPerson(p1);
    assertEquals(1, testBooks.size());
    assertEquals(0, testBooks.indexOf(b1));

    ml.checkOut(b2, p1);
    testBooks = ml.getBooksForPerson(p1);
    assertEquals(2, testBooks.size());
    assertEquals(1, testBooks.indexOf(b2));
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
public void testGetAvailableBooks() {
    setup();
    addItem();
    ArrayList<Book> testBooks = ml.getAvailableBooks();
    assertEquals(2, testBooks.size());
    assertEquals(1, testBooks.indexOf(b2));

    ml.checkOut(b1, p1);
    testBooks = ml.getAvailableBooks();
    assertEquals(1, testBooks.size());
    assertEquals(0, testBooks.indexOf(b2));

    ml.checkOut(b2, p1);
    testBooks = ml.getAvailableBooks();
    assertEquals(0, testBooks.size());
}

public void testGetUnavailableBooks() {
    setup();
    addItem();
    assertEquals(0, ml.getUnavailableBooks().size());

    ml.checkOut(b1, p1);

    ArrayList<Book> testBooks = ml.getUnavailableBooks();
    assertEquals(1, testBooks.size());
    assertEquals(0, testBooks.indexOf(b1));

    ml.checkOut(b2, p2);
    testBooks = ml.getUnavailableBooks();
    assertEquals(2, testBooks.size());
    assertEquals(1, testBooks.indexOf(b2));
}

public void testToString() {
    setup();
    addItem();
    assertEquals("Test: 2 books; 2 people.",
        ml.toString());
}
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
package org.totalbeginner.tutorial;

import java.util.ArrayList;

public class MyLibrary {

    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
        books = new ArrayList<Book>();
        people = new ArrayList<Person>();
    }

    public String getName() {
        return name;
    }

    public ArrayList<Book> getBooks() {
        return books;
    }

    public ArrayList<Person> getPeople() {
        return people;
    }

    public void addBook(Book b1) {
        this.books.add(b1);
    }

    public void removeBook(Book b1) {
        this.books.remove(b1);
    }

    public void addPerson(Person p1) {
        this.people.add(p1);
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
}

public void removePerson(Person p1) {
    this.people.remove(p1);
}

public boolean checkOut(Book b1, Person p1) {
    int booksOut = this.getBooksForPerson(p1).size();
    if ((b1.getPerson() == null) &&
        booksOut < p1.getMaximumBooks()) {
        b1.setPerson(p1);
        return true;
    }
    else {
        return false;
    }
}

public boolean checkIn(Book b1) {
    if (b1.getPerson() != null) {
        b1.setPerson(null);
        return true;
    }
    else {
        return false;
    }
}

public ArrayList<Book> getBooksForPerson(Person p1) {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if ((aBook.getPerson() != null) &&
            (aBook.getPerson().getName()
             .equals(p1.getName())))
        {
            result.add(aBook);
        }
    }
    return result;
}

public ArrayList<Book> getAvailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        if (aBook.getPerson() == null) {
            result.add(aBook);
        }
    }
    return result;
}

public ArrayList<Book> getUnavailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if (aBook.getPerson() != null) {
            result.add(aBook);
        }
    }
    return result;
}

public String toString() {
    return this.getName() + ": " +
        this.getBooks().size() + " books; " +
        this.getPeople().size() + " people.";
}
}
```

Lesson 16 – Create main Method and JAR File

```
package org.totalbeginner.tutorial;

import java.util.ArrayList;

public class MyLibrary {

    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
        books = new ArrayList<Book>();
        people = new ArrayList<Person>();
    }

    public String getName() {
        return name;
    }

    public ArrayList<Book> getBooks() {
        return books;
    }

    public ArrayList<Person> getPeople() {
        return people;
    }

    public void addBook(Book b1) {
        this.books.add(b1);
    }

    public void removeBook(Book b1) {
        this.books.remove(b1);
    }

    public void addPerson(Person p1) {
        this.people.add(p1);
    }

    public void removePerson(Person p1) {
        this.people.remove(p1);
    }

    public boolean checkOut(Book b1, Person p1) {
        int booksOut = this.getBooksForPerson(p1).size();
        if ((b1.getPerson() == null) &&
            booksOut < p1.getMaximumBooks()) {
            b1.setPerson(p1);
            return true;
        }
        else {
            return false;
        }
    }
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
public boolean checkIn(Book b1) {
    if (b1.getPerson() != null) {
        b1.setPerson(null);
        return true;
    }
    else {
        return false;
    }
}

public ArrayList<Book> getBooksForPerson(Person p1) {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if ((aBook.getPerson() != null) &&
            (aBook.getPerson().getName()
             .equals(p1.getName())))
        {
            result.add(aBook);
        }
    }
    return result;
}

public ArrayList<Book> getAvailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if (aBook.getPerson() == null) {
            result.add(aBook);
        }
    }
    return result;
}

public ArrayList<Book> getUnavailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if (aBook.getPerson() != null) {
            result.add(aBook);
        }
    }
    return result;
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
}

public String toString() {
    return this.getName() + ": " +
        this.getBooks().size() + " books; " +
        this.getPeople().size() + " people.";
}

public static void main(String[] args) {
    // create a new MyLibrary
    MyLibrary testLibrary = new MyLibrary("Test Drive Library");
    Book b1 = new Book("War And Peace");
    Book b2 = new Book("Great Expectations");
    b1.setAuthor("Tolstoy");
    b2.setAuthor("Dickens");
    Person jim = new Person();
    Person sue = new Person();
    jim.setName("Jim");
    sue.setName("Sue");

    testLibrary.addBook(b1);
    testLibrary.addBook(b2);
    testLibrary.addPerson(jim);
    testLibrary.addPerson(sue);

    System.out.println("Just created new library");
    testLibrary.printStatus();

    System.out.println("Check out War And Peace to Sue");
    testLibrary.checkOut(b1, sue);
    testLibrary.printStatus();

    System.out.println("Do some more stuff");
    testLibrary.checkIn(b1);
    testLibrary.checkOut(b2, jim);
    testLibrary.printStatus();
}

private void printStatus() {
    System.out
        .println("Status Report of MyLibrary \n" +
            this.toString());
}
```

Eclipse and Java for Total Beginners Tutorial Companion Document

```
    for (Book thisBook : this.getBooks()) {  
        System.out.println(thisBook);  
    }  
  
    for (Person p : this.getPeople()) {  
        int count = this.getBooksForPerson(p).size();  
        System.out.println(p + " (has " + count  
            + " of my books)");  
    }  
    System.out.println("Books Available: "  
        + this.getAvailableBooks().size());  
    System.out.println("--- End of Status Report ---");  
}  
}
```